A Material Point Method for Snow Simulation (Summary)

The gist of this paper is that it uses a **material point method (MPM)** to efficiently treat the wide range of material stiffnesses, collisions, and topological changes arising in complex snow scenes. MPMs combine Lagrangian material points with background Eulerian Cartesian grids. The use of the background Eulerian grid implicitly handles self-collision and fracture.

MPM relies on the *continuum approximation*, avoiding the need to model every snow grain. Although it uses the Cartesian grid to make self-collisions and topology changes automatic, it outperforms purely Eulerian simulation methods by tracking mass (and other conserved quantities) through non-dissipative Lagrangian particles. It uses the grid as an efficient continuum *scratch pad* which avoids high valence communication patterns derived from nearest-neighbor queries.

The constitutive properties central to snow are volume preservation, stiffness, plasticity, and fracture. Plasticity and fracture are handled particularly well by MPM, at the cost of slightly less accurate elasticity—a good tradeoff for snow.

The Method

A body's deformation can be described as a mapping from its undeformed configuration $\boldsymbol{X}$ to its deformed configuration $\boldsymbol{x}$:

$$\begin{aligned} \boldsymbol{x} &= \phi(\boldsymbol{X}) = \boldsymbol{F}\boldsymbol{X} + \boldsymbol{c} \\ \Rightarrow \frac{\partial \phi}{\partial \boldsymbol{X}} &= \boldsymbol{F}, \end{aligned}$$

where the 3×3 matrix $\boldsymbol{F}$ is denoted the **deformation gradient**. The deformation changes according to conservation of mass, conservation of momentum, and the elasto-plastic constitutive relation:

$$\frac{D\rho}{Dt} = 0, \quad \rho \frac{D\boldsymbol{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \rho g, \quad \boldsymbol{\sigma} = \frac{1}{J} \frac{\partial \Psi}{\partial \boldsymbol{F}_E} \boldsymbol{F}_E^T,$$

where ρ is density, t is time, $\boldsymbol{v}$ is velocity, $\boldsymbol{\sigma}$ is the Cauchy stress, g is the acceleration of gravity, Ψ is the elasto-plastic potential energy density, $\boldsymbol{F}_E$ is the elastic part of the deformation gradient, and $J = \det(\boldsymbol{F})$.

Each particle p tracks its position $\boldsymbol{x}_p$, velocity $\boldsymbol{v}_p$, mass m_p, and deformation gradient $\boldsymbol{F}_p$. The Lagrangian treatment of these quantities simplifies the discretization of the $\frac{D\rho}{Dt}$ and $\rho\frac{D\boldsymbol{v}}{Dt}$ terms. However, the lack of mesh connectivity between particles makes it harder to compute the derivatives needed for stress-based force evaluation.

In order to remedy this, interpolating functions are used to transfer the terms needed for the computation of $\nabla \cdot \boldsymbol{\sigma}$ from the particles to the regular background grid. Define the function $N(x)$ as

$$N(x) = \begin{cases} \frac{1}{2}|x|^3 - x^2 + \frac{2}{3}, & 0 \le |x| < 1 \\ -\frac{1}{6}|x|^3 + x^2 - 2|x| + \frac{4}{3}, & 1 \le |x| < 2 \\ 0, & \text{otherwise} \end{cases}.$$

This is a one-dimensional B-spline curve, nonzero on $x \in [-2, 2]$. The function is shown plotted against x in Figure 1 on the left.

This function will be used in three dimensions to weight the contribution of a particle's quantity to a node on the regular grid. Specifically, consider a node $\boldsymbol{i} = (i, j, k)$ in a grid with uniform spacing h, and an evaluation position $\boldsymbol{x_p} = (x_p, y_p, z_p)$. The grid basis function is defined as

$$N_{\boldsymbol{i}}^h(\boldsymbol{x_p}) = N(\tfrac{1}{h}(x_p - ih))N(\tfrac{1}{h}(y_p - jh))N(\tfrac{1}{h}(z_p - kh)).$$

For a more compact notation, the interpolation weights are denoted as $w_{\boldsymbol{i}p} = N_{\boldsymbol{i}}^h(\boldsymbol{x}_p)$. These interpolation weights are used to transfer the mass and momentum from the particles to the grid so that the velocities can

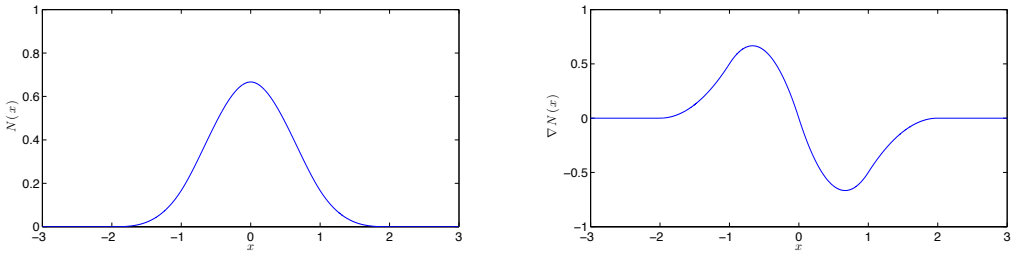

Figure 1: The one-dimensional interpolation weight function and its gradient.

be updated at the grid nodes. The important consequence of this weighting function is that only particles within a radius of $2h$ contribute to the quantities at any grid node.

We'll also need the gradient of these interpolation weights, $\nabla w_{ip} = \nabla N_i^h(\boldsymbol{x}_p)$. If we define $d = |x|$—i.e., the distance from $x = 0$, then for $d \in [0, 2]$

$$N(d) = \begin{cases} \frac{1}{2}d^3 - d^2 + \frac{2}{3}, & 0 \le d < 1 \\ -\frac{1}{6}d^3 + d^2 - 2d + \frac{4}{3}, & 1 \le d < 2 \end{cases} \Rightarrow N_d(d) = \begin{cases} \frac{3}{2}d^2 - 2d, & 0 \le d < 1 \\ -\frac{1}{2}d^2 + 2d - 2, & 1 \le d < 2 \end{cases}$$

where $N_d(d)$ is the partial derivative of $N(d)$ with respect to d.

If $x \ge 0$, we use $N_x(x) = N_d(d)$; if $x < 0$, we use $N_x(x) = -N_d(d)$. The one-dimensional gradient of this interpolation function is shown plotted against x in Figure 1 on the right.

This allows us to express the gradient of the interpolation weights:

$$\nabla w_{ip} = \nabla N_i^h(\boldsymbol{x}_p) = \left(N_{ix}^h(\boldsymbol{x}_p), N_{iy}^h(\boldsymbol{x}_p), N_{iz}^h(\boldsymbol{x}_p)\right)^T,$$

where

$$\begin{aligned} N_{ix}^h(\boldsymbol{x}_p) &= N_x(\tfrac{1}{h}(x_p - ih))N(\tfrac{1}{h}(y_p - jh))N(\tfrac{1}{h}(z_p - kh)), \\ N_{iy}^h(\boldsymbol{x}_p) &= N(\tfrac{1}{h}(x_p - ih))N_x(\tfrac{1}{h}(y_p - jh))N(\tfrac{1}{h}(z_p - kh)), \\ N_{iz}^h(\boldsymbol{x}_p) &= N(\tfrac{1}{h}(x_p - ih))N(\tfrac{1}{h}(y_p - jh))N_x(\tfrac{1}{h}(z_p - kh)). \end{aligned}$$

The full method is outline below. We have the quantities for every particle $p \in P$ at time t^n, and wish to step the simulation forward by Δt to time t^{n+1}.

1. **Rasterize particle data to the grid.** First, the mass is transferred from the points to the grid using the interpolation weights:

$$m_i^n = \sum_{p \in P} m_p w_{ip}^n.$$

To transfer velocity while conserving momentum, we normalize the result by m_i^n:

$$\boldsymbol{v}_i^n = \frac{1}{m_i^n} \sum_{p \in P} \boldsymbol{v}_p^n m_p w_{ip}^n.$$

2. **Compute particle volumes and densities.** The force discretization requires a notion of the particle's volume in its initial configuration, so we *only need to perform this step for the first time step*, for

$n = 0$. A grid cell's density is just its mass divided by h^3. We can weight these back to the particles as:

$$\rho_p^0 = \frac{1}{h^3} \sum_{\boldsymbol{i}} m_{\boldsymbol{i}}^0 w_{\boldsymbol{i}}^0.$$

With this, a particle's volume can be estimated as

$$V_p^0 = \frac{m_p}{\rho_p^0}.$$

3. Compute grid forces.

(i) The deformation gradient $\boldsymbol{F}$ is separated into an elastic part $\boldsymbol{F}_E$ and a plastic part $\boldsymbol{F}_P$, where $\boldsymbol{F} = \boldsymbol{F}_E \boldsymbol{F}_P$. Initially, since the system is in its undeformed state, these elastic and plastic components of the deformation gradient are simply identity. They are subsequently updated separately during each time step.

(ii) In linear elasticity, the **Lamé parameters** are $\boldsymbol{\lambda}$ (the first Lamé parameter), and $\boldsymbol{\mu}$ (the second Lamé parameter, often called the shear modulus). Given the paper's suggested parameters for initial Young's modulus, E_0, and Poission's ratio, ν, these Lamé parameters can be computed as

$$\lambda_0 = \frac{E_0 \nu}{(1+\nu)(1-2\nu)} \quad \text{and} \quad \mu_0 = \frac{E_0}{2(1+\nu)}.$$

(iii) The constitutive model is defined in terms of the elasto-plastic energy density function

$$\Psi\left(\boldsymbol{F}_E, \boldsymbol{F}_P\right) = \mu\left(\boldsymbol{F}_P\right) \left\|\boldsymbol{F}_E - \boldsymbol{R}_E\right\|_{\boldsymbol{F}}^2 + \frac{\lambda\left(\boldsymbol{F}_P\right)}{2} \left(J_E - 1\right)^2,$$

where the Lamé parameters above are functions of the plastic deformation gradients,

$$\mu\left(\boldsymbol{F}_P\right) = \mu_0 e^{\xi\left(1-J_P\right)} \quad \text{and} \quad \lambda\left(\boldsymbol{F}_P\right) = \lambda_0 e^{\xi\left(1-J_P\right)},$$

$J_E = \det\left(\boldsymbol{F}_E\right)$ and $J_P = \det\left(\boldsymbol{F}_P\right)$; $\boldsymbol{R}_E$ is the orthonormal matrix of the polar decomposition of $\boldsymbol{F}_E = \boldsymbol{R}_E \boldsymbol{S}_E$; λ_0 and μ_0 are the initial Lamé coefficients; and ξ is a constant plastic hardening parameter.

(iv) The force on grid node $\boldsymbol{i}$ resulting from elastic stresses can be expressed

$$\boldsymbol{f}_{\boldsymbol{i}}(\boldsymbol{x}) = -\sum_{p \in P} V_p^n \boldsymbol{\sigma}_p \nabla w_{\boldsymbol{i}p}^n,$$

where $V_p^n = J_p^n V_p^0$ is the volume of the material occupied by particle p at time n and $\boldsymbol{\sigma}_p$ is the Cauchy stress:

$$\boldsymbol{\sigma}_p = \frac{1}{J_p^n} \frac{\partial}{\partial \boldsymbol{F}_E} \Psi\left(\boldsymbol{F}_{Ep}^n, \boldsymbol{F}_{Pp}^n\right) \left(F_{Ep}^n\right)^T$$

$$= \frac{2\mu\left(\boldsymbol{F}_{Pp}^n\right)}{J_p^n} \left(\boldsymbol{F}_{Ep}^n - \boldsymbol{R}_{Ep}^n\right) \left(\boldsymbol{F}_{Ep}^n\right)^T + \frac{\lambda\left(\boldsymbol{F}_{Pp}^n\right)}{J_p^n} \left(J_{Ep}^n - 1\right) J_{Ep}^n \boldsymbol{I}$$

4. Update grid velocities. Using the estimated grid forces, we update the grid velocities to $\boldsymbol{v}_{\boldsymbol{i}}^*$ using:

$$\boldsymbol{v}_{\boldsymbol{i}}^* = \boldsymbol{v}_{\boldsymbol{i}}^n + \Delta t \frac{\boldsymbol{f}_{\boldsymbol{i}}^n}{m_{\boldsymbol{i}}^n}.$$

5. **Grid-based body collisions.** Collision objects can be represented as level sets, such that collisions are trivial to detect, when $\phi(\boldsymbol{x_i}) \leq 0$. When collisions are detected, the collision object's normal $\boldsymbol{n} = \nabla\phi(\boldsymbol{x_i})$ and its velocity $\boldsymbol{v}_{co}$ are computed.

   The grid velocity is first transformed to the relative frame of the collision object

   $$\boldsymbol{v}_{rel} = \boldsymbol{v_i^*} - \boldsymbol{v}_{co}.$$

   The magnitude of the normal component of the relative velocity is computed as

   $$v_n = \boldsymbol{v}_{rel} \cdot \boldsymbol{n}.$$

   If the bodies are separating—i.e., $v_n \geq 0$—no collision is applied. If not, the tangential velocity is computed as

   $$\boldsymbol{v}_t = \boldsymbol{v}_{rel} - \boldsymbol{n}v_n.$$

   From here, there are two cases. Where μ is the coefficient of friction, if

   $$\|\boldsymbol{v}_t\| \leq -\mu v_n,$$

   the tangential component is insignificant compared to the normal component, and we simply set $\boldsymbol{v}'_{rel} = 0$, where the prime indicates that collision has been applied. Otherwise, we apply dynamic friction, and

   $$\boldsymbol{v}'_{rel} = \boldsymbol{v}_t + \frac{\mu v_n}{\|\boldsymbol{v}_t\|}\boldsymbol{v}_t = \left(1 + \frac{\mu v_n}{\|\boldsymbol{v}_t\|}\right)\boldsymbol{v}_t$$

   Finally, we transform the velocity back to the world reference frame

   $$\boldsymbol{v_i^{*\prime}} = \boldsymbol{v}'_{rel} + \boldsymbol{v}_{co}.$$

6. **Integration.** For semi-implicit integration, solve the complicated linear system described in the paper; for explicit integration, simply let $\boldsymbol{v_i}^{n+1} = \boldsymbol{v_i^*}$.

7. **Update the deformation gradient components.**

   (i) We define the portion of deformation that is elastic and plastic using the singular values of the deformation gradient. Recall that the singular value decomposition of the matrix $\boldsymbol{F}$ is a factorization of the form

   $$\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}},$$

   where $\boldsymbol{U}$ and $\boldsymbol{V}$ are both orthonormal (rotation) matrices, and $\boldsymbol{\Sigma}$ is a diagonal matrix with nonnegative real numbers, whose entries are called the **singular values** of $\boldsymbol{F}$. We define a critical compression θ_c and stretch θ_s as the thresholds on these singular values beyond which to start plastic deformation. In other words, the singular values of the elastic component of the deformation gradient, $\boldsymbol{F}_E$, are restricted to $[1 - \theta_c, 1 + \theta_s]$; any deformation beyond this is considered plastic.

   (ii) The new deformation gradient is

   $$\boldsymbol{F}_p^{n+1} = \left(\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}_p^{n+1}\right)\boldsymbol{F}_{Ep}^n\boldsymbol{F}_{Pp}^n.$$

   Begin by temporarily defining

   $$\hat{\boldsymbol{F}}_{Ep}^{n+1} = \left(\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}_p^{n+1}\right)\boldsymbol{F}_{Ep}^n \quad \text{and} \quad \hat{\boldsymbol{F}}_{Pp}^{n+1} = \boldsymbol{F}_{Pp}^n.$$

   What's happening here is that we're temporarily assigning all the changes to the elastic part of the deformation gradient.

(iii) Now we want to take all the deformation that exceeds the critical deformation thresholds and push it into $\hat{\boldsymbol{F}}_{Pp}^{n+1}$. Compute the singular value decomposition of the temporary elastic component:

$$\hat{\boldsymbol{F}}_{Ep}^{n+1} = \boldsymbol{U}_p \hat{\boldsymbol{\Sigma}}_p \boldsymbol{V}_p^T$$

and clamp the singular values to the permitted range:

$$\boldsymbol{\Sigma}_p = \text{clamp}\left(\hat{\boldsymbol{\Sigma}}_p,\ [1-\theta_c, 1+\theta_s]\right)$$

(iv) The final elastic component of the deformation gradient is

$$\boldsymbol{F}_{Ep}^{n+1} = \boldsymbol{U}_p \boldsymbol{\Sigma}_p \boldsymbol{V}_p^T,$$

and the final plastic component of the deformation gradient is

$$\boldsymbol{F}_{Pp}^{n+1} = \boldsymbol{V}_p \boldsymbol{\Sigma}_p^{-1} \boldsymbol{U}_p^T \boldsymbol{F}_p^{n+1}$$

8. Update particle velocities. To transfer the grid velocities back to the particles, a combination of **particle-in-cell (PIC)** and **fluid-implicit-particle (FLIP)** manners is used:

$$\boldsymbol{v}_p^{n+1} = (1-\alpha)\boldsymbol{v}_{\text{PIC}p}^{n+1} + \alpha \boldsymbol{v}_{\text{FLIP}p}^{n+1},$$

where

$$\boldsymbol{v}_{\text{PIC}p}^{n+1} = \sum_{\boldsymbol{i}} \boldsymbol{v}_{\boldsymbol{i}}^{n+1} w_{\boldsymbol{i}p}^n$$

and

$$\boldsymbol{v}_{\text{FLIP}p}^{n+1} = \boldsymbol{v}_p^n + \sum_{\boldsymbol{i}} \left(\boldsymbol{v}_{\boldsymbol{i}}^{n+1} - \boldsymbol{v}_{\boldsymbol{i}}^n\right) w_{\boldsymbol{i}p}^n$$

with a typical value chosen to be $\alpha = 0.95$.

9. Particle-based body collisions. The process for particle-based body collisions is the same as for grid-based body collisions, just use

$$\boldsymbol{v}_{rel} = \boldsymbol{v}_p^{n+1} - \boldsymbol{v}_{co}.$$

10. Update particle positions. Using the updated particle velocities, move the particles:

$$\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \Delta t \boldsymbol{v}_p^{n+1}.$$